Capabilities of Constraint Programming in Safe Global Optimization*

Alexandre Goldsztejn✠, Yahia Lebbah‡,†, Claude Michel† and Michel Rueher†

† Université de Nice Sophia Antipolis, CNRS, 06903 Sophia Antipolis, France,
✠ CNRS, Université de Nantes, 44322 Nantes, France
‡ Université d'Oran Es-Senia B.P. 1524 EL-M'Naouar, 31000 Oran,Algeria

Abstract

We investigate the capabilities of constraints programming techniques in rigorous global optimization methods. We introduce different constraint programming techniques to reduce the gap between efficient but unsafe systems like Baron[1], and safe but slow global optimization approaches. We show how constraint programming filtering techniques can be used to implement optimality-based reduction in a safe and efficient way, and thus to take advantage of the known bounds of the objective function to reduce the domain of the variables, and to speed up the search of a global optimum. We describe an efficient strategy to compute very accurate approximations of feasible points. This strategy takes advantage of the Newton method for under-constrained systems of equalities and inequalities to compute efficiently a promising upper bound. Experiments on the COCONUT benchmarks demonstrate that these different techniques drastically improve the performances.

1 Introduction

We consider here the global optimization problem $\mathcal{P}$ to minimize an objective function under nonlinear equalities and inequalities,

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) = 0,\ i \in \{1,..,k\} \\ & h_j(x) \leq 0,\ j \in \{1,..,m\} \end{array} \tag{1}$$

with $x \in \mathbf{x}$, $f : \boldsymbol{R}^n \to \boldsymbol{R}$, $g_i : \boldsymbol{R}^n \to \boldsymbol{R}$ and $h_j : \boldsymbol{R}^n \to \boldsymbol{R}$; Functions f , g_i and h_j are nonlinear and continuously differentiable on some vector $\mathbf{x}$ of intervals of $\boldsymbol{R}$. For convenience, in the following text, $g(x)$ (resp. $h(x)$) will denote the vector of $g_i(x)$ (resp. $h_j(x)$) functions.

*This paper is an extended version of [17]. Preliminary results have been published in [10] and [5].

†This work was partially supported by the European Community's 7th Framework Programme (FP7/2007-2013), MANCOOSI project, grant agreement n. 214898.

[1] See http://www.andrew.cmu.edu/user/ns1b/baron/baron.html

The difficulties in such global optimization problems come mainly from the fact that many local minimizers may exist but only few of them are global minimizers [14].

Optimality-based reduction (OBR) has been introduced by Ryoo and Sahinidis in [18] to take advantage of the known bounds of the objective function to reduce the size of the domains of the variables. This technique uses a well known property of the saddle point to compute new bounds for the domain of a variable which take into account the known bounds of the objective function. However, the basic OBR algorithm is unsafe.

Kearfott [7, 6] has proposed a safe implementation of OBR which is based on a valid bounding of the dual solution. Note that this method suffers from strong limitations and is rather slow.

We show that constraint programming techniques can be used in a simple and elegant way to safely refute the potential non-solution boxes identified by the OBR method. Roughly speaking, filtering techniques are used to reduce these boxes to empty boxes, and thus, to prove that they do not contain any feasible point. These constraint programming techniques do not suffer from the same limitations as Kearfott's method. Experiments show that they are also much more efficient.

In global optimization problems, the feasible region may be disconnected. Thus, finding feasible points is a critical issue in safe Branch and Bound algorithms for continuous global optimization. Standard strategies use local search techniques to provide a reasonable approximation of an upper bound and try to prove that a feasible solution actually exists within the box built around the guessed global optimum. Practically, finding a guessed point for which the proof succeeds is often a very costly process.

We introduce a new strategy to compute very accurate approximations of feasible points. This strategy takes advantage of the Newton method for under-constrained systems of equalities and inequalities. More precisely, this procedure exploits the optimal solution of a linear relaxation of the problem to compute efficiently a promising upper bound. Experiments on the COCONUT benchmarks demonstrate that the combination of this procedure with a safe branch and bound algorithm drastically improves the performances.

The rest of this paper is organized as follows. The first section provides the overall schema of a safe branch and bound process for global optimization. The next section describes the OBR method and introduces our safe implementation based on constraint techniques. Next, we describe our new strategy to compute very accurate approximations of feasible points

We do not recall here the capabilities of consistency techniques to speed up the initial convergence of the interval narrowing algorithms. Neither do we show how linear relaxations can be used in such a CP framework to rigorously bound the global optima as well as its location. A detailed discussion of these concepts and techniques can be found in [9, 11, 19].

2 The Branch and Bound schema

The algorithms we describe here are derived from the well known Branch and Bound schema introduced by Horst and Tuy for finding a global minimizer. Interval analysis techniques are used to ensure rigorous and safe computations whereas

Algorithm 1 Branch and Bound Algorithm

Function BB(IN $\mathbf{x}$, ϵ; OUT $\mathcal{S}$, $[L, U]$)

% $\mathcal{S}$: set of proven feasible points
% $\mathbf{f}_{\mathbf{x}}$ denotes the set of possible values for f in $\mathbf{x}$
% *nbStarts*: number of starting points in the first upper-bounding
$\mathcal{L} \leftarrow \{\mathbf{x}\}$; $(L, U) \leftarrow (-\infty, +\infty)$;
$\mathcal{S} \leftarrow UpperBounding(\mathbf{x}, nbStarts)$;
while $w([L, U]) > \epsilon$ **do**
  $\mathbf{x}' \leftarrow \mathbf{x}''$ such that $\underline{\mathbf{f}}_{\mathbf{x}''} = min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}$;
  $\mathcal{L} \leftarrow \mathcal{L} \setminus \{\mathbf{x}\}'$;
  $\bar{\mathbf{f}}_{\mathbf{x}'} \leftarrow min(\bar{\mathbf{f}}_{\mathbf{x}'}, U)$;
  $\mathbf{x}' \leftarrow Prune(\mathbf{x}')$;
  $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$;
  $\mathcal{S} \leftarrow \mathcal{S} \cup UpperBounding(\mathbf{x}', 1)$;
  if $\mathbf{x}' \neq \emptyset$ **then**
    $(\mathbf{x}'_1, \mathbf{x}'_2) \leftarrow Split(\mathbf{x}')$;
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathbf{x}'_1, \mathbf{x}'_2\}$;
  endif
  if $\mathcal{L} = \emptyset$ **then**
    $(L, U) \leftarrow (+\infty, -\infty)$;
  else
    $(L, U) \leftarrow (min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{L}\}, min\{\bar{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathcal{S}\})$;
  endif
endwhile

constraint programming techniques are used to improve the reduction of the feasible space.

Algorithm 1 computes enclosers for minimizers and safe bounds of the global minimum value within an initial box $\mathbf{x}$. Algorithm 1 maintains two lists : a list $\mathcal{L}$ of boxes to be processed and a list $\mathcal{S}$ of proven feasible boxes. It provides a rigorous encloser $[L, U]$ of the global optimum with respect to a tolerance ϵ.

Algorithm 1 starts with $UpperBounding(\mathbf{x}, nbStarts)$ which computes a set of feasible boxes by calling a local search with *nbStarts* starting points and a proof procedure. The boxes built around the local solution are added to $\mathcal{S}$ if and only if they are proved to contain a feasible point.

In the main loop, Algorithm 1 selects the box $\mathbf{x}'$ with the lowest lower bound of the objective function. The *Prune* function applies filtering techniques to reduce the size of the box $\mathbf{x}'$. In the framework we have implemented, *Prune* just uses a 2B-filtering algorithm [13]. Then, $LowerBound(\mathbf{x}')$ computes a rigorous lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ using a linear programming relaxation of the initial problem. Actually, function *LowerBound* is based on the linearization techniques of the `Quad` framework [11]. *LowerBound* computes a safe minimizer $\underline{\mathbf{f}}_{\mathbf{x}'}$ thanks to the techniques introduced by Neumaier et al.

Algorithm 1 then calls the upper bounding procedure on box $\mathbf{x}'$. Here again, the box around the local solution is added to $\mathcal{S}$ if it is proved to contain a feasible point. At this stage, if the box $\mathbf{x}'$ is empty then, either it does not contain any feasible point or its lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ is greater than the current upper bound U. If $\mathbf{x}'$ is not empty, the box is split along one of the variables of the problem. Various

heuristics are used to select the variable the domain of which has to be split.

Algorithm 1 maintains the lowest lower bound L of the remaining boxes $\mathcal{L}$ and the lowest upper bound U of proven feasible boxes. The algorithm terminates when the space between U and L becomes smaller than the given tolerance ϵ. Of course a proven optimum cannot always be found, and thus, algorithm 1 has to be stopped in some cases to get the feasible boxes which may have been found.

The next section is devoted to OBR techniques. We first recall the basic definitions, then we describe the method proposed by Kearfott, and finally we introduce our safe algorithm for computing OBR.

3 Optimality-based reduction

3.1 Basics of optimality-based reduction

Optimality-based reduction has been introduced by Ryoo and Sahinidis in [18]. It takes advantage of a property of the saddle point to reduce the domains of the variables of the optimization problem. Optimality-based reduction relies on the following two theorems to improve the bounds of the domain of one variable, so we assume that the variable x_i is bounded inside $[\underline{x}_i, \overline{x}_i]$ (which is equivalent to the two linear constraints $x_i - \overline{x}_i \leq 0$ and $\underline{x}_i - x_i \leq 0$).

Theorem 1. *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $x_i - \overline{x}_i \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$x_i \geq x_i' \text{ with } x_i' = \overline{x}_i - \frac{U - L}{\lambda_i^*}. \tag{2}$$

Thus, if $x_i' > \underline{x}_i$, the domain of x_i can be set to $[x_i', \overline{x}_i]$ without loss of any global optima.

λ_i^* denotes the dual solution of R. A convex relaxation of a minimization problem P is a convex minimization problem (i.e. the cost and the constraints are convex) whose feasible set is larger than the original one, and whose cost function is lower than the original one. Thus, the minimum of the relaxation is lower than the minimum of P. A constraint $g(x) \leq 0$ is active if $g(x) = 0$. Such an equality may be difficult to be checked over the floating-point numbers.

Theorem 2. *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $\underline{x}_i - x_i \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$x_i \leq x_i'' \text{ with } x_i'' = \underline{x}_i + \frac{U - L}{\lambda_i^*}. \tag{3}$$

Thus, if $x_i'' < \overline{x}_i$, the domain of x_i can be set to $[\underline{x}_i, x_i'']$ without loss of any global optima.

The first theorem provides a test to improve the lower bound of the domain of a variable while the second theorem provides a test to improve the upper bound of the domain of a variable.

Moreover, these valid inequalities have been generalized to the other constraints. The following theorem is the most general one :

Theorem 3. *Let U be a known upper bound of the original problem P, let L be a known lower bound of a convex relaxation R of P, and assume that the constraint $g_i(x) \leq 0$ is active at the optimal solution of R and has a corresponding multiplier $\lambda_i^* > 0$. Then*

$$g_i(x) \geq -\frac{U-L}{\lambda_i^*}. \tag{4}$$

This last theorem enables to enforce some constraints, and thus to reduce the domains of the variables.

All these theorems are explained in detail and proved in [18].

3.2 Kearfott's approach: Safe OBR based on bounding rigorously dual variables

The critical issue in optimality-based reduction formulae (2–4) comes from the unsafe dual solution provided by the Simplex method. Note that the techniques suggested by Neumaier et al [15] and used in the `Quad` framework cannot be applied to compute a safe solution of the dual problem. So, the validity of dual solutions must be proved to keep the branch and bound process rigorous. This approach has been investigated by Kearfott [6] in the specific case of linear relaxations. For the sake of simplicity, we consider the following linear relaxation formulated with lower inequalities:

$$\begin{array}{ll} min & d^T x \\ s.t. & Ax \leq b \end{array} \tag{5}$$

The dual of (5) is the following LP

$$\begin{array}{ll} max & b^T y \\ s.t. & A^T \lambda = d \end{array} \tag{6}$$

where λ denotes the dual variables required in OBR formulae (2)(3)(4). The Kuhn-Tucker system (KT)[2] is used to provide validated lower and upper bounds on the system (5) and (6).

$$(KT) \left\{ \begin{array}{ll} A^T \lambda - d & = 0 \\ \lambda_i (A_{i,:} x - b_i) & = 0, 1 \leq i \leq m \end{array} \right. \tag{7}$$

where $A_{i,:}$ is i-th row of A.

It is well known that if the i-th constraint $A_{i,:} x \leq b_i$ is inactive for some solution x^* –i.e., strict inequality $A_{i,:} x^* < b_i$ holds for x^*, the solution of the primal– then the corresponding dual variable (also called Lagrange multiplier) y_i is equal to zero. Thus, inactive constraints must be identified to make the whole system (7) linear. In this approach there are two main critical issues:

1. Some of the constraints of (7) are redundant due to the fact that each equality constraint is replaced by two inequality constraints. So, the Newton methods cannot be used for the validation process.
2. Inactive constraints are identified with the approximate dual solution provided by the Simplex method (i.e., inactive constraints have a null dual solution).

[2]For a detailed introduction to Kuhn-Tucker system, see [1].

Kearfott handles these issues by weakening the relaxation. Consequently, the final validation method based on the Newton method applied to (7) does not always succeed and, when it succeeds, the bounds could be wide due to the fact that the relaxation has been weakened.

3.3 A safe implementation of OBR based on constraint filtering techniques

As said before, the critical issue in the OBR method comes from the unsafe dual solution provided by the simplex algorithm. In other words, due to the rounding errors, we may lose the global optima when we use formula (2) to shrink the domain of some variable x_i.

The essential observation is that we can use filtering techniques to prove that no feasible point exists when the domain of x_i is reduced to $[\underline{x_i}, x_i']$. Indeed, if the constraint system

$$\begin{array}{l} f(x) \leq U \\ g_i(x) = 0, \ \ i = 1..k \\ g_j(x) \leq 0, \ \ j = k+1..m \end{array} \tag{8}$$

does not have any solution when the domain of x is set to $[\underline{x_i}, x_i']$, then the domain reduction computed by the OBR method is valid; if the filtering cannot prove that no solution exists inside the considered box, we have just to add this box to $\mathcal{L}$, the list of boxes to be processed (See algorithms 1 and 2).

The same reasoning holds for the reduction of the domain of $\mathbf{f}$ (see [10]), i.e., when algorithm 2 attempts to reduce the upper bound of the variables of the problem by means of formula (3).

Algorithm 2 details the new process of the computation of the lower bound. Note that Algorithm 1 remains almost unchanged : we have just to replace the call $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow$ *LowerBound*($\mathbf{x}'$) by ($\underline{\mathbf{f}}_{\mathbf{x}'}, x', \mathcal{L}$)$\leftarrow$*LowerBound*($\mathbf{x}', L, U, \mathcal{L}$).

The constraint-based approach introduced here is about five time faster than Kearfott's approach. In fact, our approach introduces a negligible overhead since the proof process mostly relies on a $2B$-consistency which is an effective technique here. That is why the more costly *Quad*-filtering (see [12]) is almost never used in these examples.

Next section is devoted to the new method we propose to compute efficiently a promising upper bound.

4 A new upper bounding strategy

The standard upper bounding procedure relies on a local search to provide a "guessed" feasible point lying in the neighborhood of a local optima. However, the effects of floating point computation on the provided local optima are hard to predict. As a result, the local optima might lie outside the feasible region and the proof procedure might fail to prove the existence of a feasible point within the box built around this point.

We propose here a new upper bounding strategy which attempts to take advantage of the solution of a linear outer approximation of the problem. The lower bound process uses such an approximation to compute a safe lower bound of $\mathcal{P}$. When the LP is solved, a solution x_{LP} is always computed and, thus, available for

Algorithm 2 Computation of a safe lower bound with OBR

Function `LowerBound`(`IN` **x**, L, U,$\mathcal{L}$; `OUT` ($\underline{\mathbf{f}}_{\mathbf{x}'}$,$\mathcal{L}$))

$\mathcal{L}_r \leftarrow \emptyset$ % $\mathcal{L}_r$: set of potential non-solution boxes;
Compute $\underline{\mathbf{f}}$ with `Quad` in **x**;
for each variable **x do**
  Apply formula 2 of OBR using L, U;
  Add the generated potential non-solution boxes to $\mathcal{L}_r$;
endfor
for each box $\mathbf{B_i}$ in $\mathcal{L}_r$ **do**
  $\mathbf{B'_i} \leftarrow 2B$-filtering($\mathbf{B_i}$);
  if $\mathbf{B'_i} = \emptyset$ **then**
    Reduce the domain of x_i;
  else
    $\mathbf{B''_i} \leftarrow$ `Quad`-filtering($\mathbf{B'_i}$);
    if $\mathbf{B''_i} = \emptyset$ **then**
      Reduce the domain of x_i;
    else
      Add $\mathbf{B''_i}$ to $\mathcal{L}$;
    endif
  endif
endfor
Apply formula (3) of OBR to reduce the lower bound of the variables;
Use $2B$-filtering and `Quad`-filtering to validate the reduction;

free. This solution being an optimal solution of an outer approximation of $\mathcal{P}$, it lies outside the feasible region. Thus, x_{LP} is not a feasible point. Nevertheless, x_{LP} may be a good starting point to consider for the following reasons:

- At each iteration, the branch and bound process splits the domain of the variables. The smaller the domain is, the closer x_{LP} is to the actual optima of $\mathcal{P}$.
- The proof process inflates a box around the initial guess. This process may compensate the effect of the distance of x_{LP} from the feasible region.

However, while x_{LP} converges to a feasible point, the process might be quite slow. To speed up the upper bounding process, we have introduced a lightweight, though efficient, procedure which computes a feasible point from a point lying in the neighborhood of the feasible region. This procedure which is called *FeasibilityCorrection* will be detailed in the next subsection.

Algorithm 3 describes how an upper bound may be build from the solution of the linear problem used in the lower bounding procedure.

4.1 Computing pseudo-feasible points

This section introduces an adaptation of the Newton method to under-constrained systems of equalities and inequalities which provides very accurate approximations of feasible points at a low computational cost[3]. Discovering feasible points is

[3] A similar method was briefly mentioned though not described in [3].

Algorithm 3 Upper bounding build from the LP optimal solution x^*_{LP}

Function `UpperBounding(IN` $\mathbf{x}$, x^*_{LP}, $nbStarts$; `OUT` $\mathcal{S}'$`)`

% $\mathcal{S}'$: list of proven feasible boxes
% $nbStarts$: number of starting points
% x^*_{LP}: the optimal solution of the LP relaxation of $\mathcal{P}(\mathbf{x})$
$\mathcal{S}' \leftarrow \emptyset$;
$x^*_{corr} \leftarrow$FeasibilityCorrection(x^*_{LP});
$\mathbf{x}_p \leftarrow$InflateAndProve(x^*_{corr}, $\mathbf{x}$);
if $\mathbf{x}_p \neq \emptyset$ **then** $\mathcal{S}' \leftarrow \mathcal{S}' \cup \mathbf{x}_p$;
return $\mathcal{S}'$

an important step of branch and bound algorithms. That is why a detailed setting of this method is presented below.

Given an approximate solution $x_k \in \mathbf{R}^n$ to a system of equalities $g(x) = 0$, with $g : \mathbf{R}^n \longrightarrow \mathbf{R}^m$, a step of the Newton method consists of finding a better approximate solution $x_{k+1} := x_k + h$, with $h \in \mathbf{R}^n$, by solving the linearization of $g(x_k + h) = 0$. That is, the displacement h between step k and $k+1$ is chosen as the solution of the linear equality

$$g(x_k) + Dg(x_k)\, h = 0, \tag{9}$$

where $Dg(x) \in \mathbf{R}^{m \times n}$ is the Jacobian of g evaluated at x. When the system of equalities $g(x) = 0$ is under-constrained, i.e. $m < n$, it has a manifold of solutions and the linearized equality (9) has an affine subspace of solutions. Since the linearization (9) is accurate in a neighborhood of x_k, it is natural to select among its affine solution set the solution that minimizes the distance to x_k. This is done using the Moore-Penrose inverse of $Dg(x_k)$ [2], denoted by $Dg(x_k)^+$. We obtain the following Newton iteration for under-constrained systems of equalities:

$$x_{k+1} := x_k - Dg(x_k)^+\, g(x_k). \tag{10}$$

The Moore-Penrose inverse of $A \in \mathbf{R}^{m \times n}$ can be computed in several ways: using a singular value decomposition, or by means of the formula $A^+ = A^T(AA^T)^{-1}$ provided that A is full (row) rank.

The nice property of this Newton method is that it converges quadratically for under-constrained systems of equalities:

Proposition 1. *Suppose that the sequence x_k defined by (10) converges toward x_* and assume that the step k error ϵ_k is defined by $x_k = x_* + \epsilon_k$. Then $\epsilon_{k+1} = O(\epsilon_k)^2$, that is ϵ_k converges quadratically toward zero. In other words, the number of correct decimals is roughly multiplied by two at each new iteration.*

Proof. Using a first order expansion $g(x_k) = g(x_*) + Dg(x_*)\epsilon_k + O(\epsilon_k)^2 = Dg(x_*)\epsilon_k + O(\epsilon_k)^2$, and a zero order expansion $Dg(x_k)^+ = Dg(x_*)^+ + O(\epsilon_k)$. Now, $\epsilon_{k+1} - \epsilon_k = x_{k+1} - x_k$, the latter being equal to $-Dg(x_k)^+\, g(x_k)$ by (10). Using the first two expansions we obtain

$$\epsilon_{k+1} - \epsilon_k = -(Dg(x_*)^+ + O(\epsilon_k))(Dg(x_*)\epsilon_k + O(\epsilon_k)^2).$$

	$\Sigma_t(s)$	$\%saving$
no OBR	2384.36	-
unsafe OBR	881.51	63.03%
safe OBR Kearfott	1975.95	17.13%
safe OBR CP	454.73	80.93%

Table 1: Synthesis of the results on 78 benchmarks (with a timeout of 500s)

Expanding the right hand side term, we obtain $\epsilon_{k+1} - \epsilon_k = -Dg(x_*)^+ Dg(x_*)\epsilon_k + O(\epsilon_k)^2$. Note that A^+A might be different from I, but by (10), we have $\epsilon_k = -Dg(x_k)^+ g(x_k)$, and, since $A^+AA^+ = A^+$, we finally obtain $\epsilon_{k+1} - \epsilon_k = -\epsilon_k + O(\epsilon_k)^2$, which concludes the proof. □

Inequality constraints can be changed to equalities by introducing slack variables: $h_j(x) \leq 0 \iff h_j(x) = -s_i^2$. So, the Newton method for under-constrained systems of equalities can also be applied to systems of equalities and inequalities.

5 Experiments

This section reports the results of experiments done with various combinations of the aforementioned strategies on a significant set of benchmarks.

All the benchmarks come from the collection of benchmarks of the Coconuts project[4]. Note that all the problems are detailed on the Coconuts website. From this library, benchmarks with more than 100 variables or 100 constraints[5] have been rejected, as well as benchmarks using functions that are not yet handled by Icos, like power with real numbers (i.e. x^y where y is not a positive integer) or binary variables. Obviously, Icos handles x^n where n is a positive integer. Due to the lack of space, we report here only a subset of 50 selected benchmarks.

All the tests have been ran with Icos[6] [8] on an Intel Xeon X5460 under linux. A time out of 60s is used to bound the running time (a "-" appear in the columns when this time out is reached). Icos relies on Ipopt for the local search and on COIN CLP to solve the linear relaxation. The proof of existence uses an implementation of the Borsuk theorem [4].

Table 1 presents a synthesis of a comparison of various OBR implementations done on a set of 78 benchmarks (see [10] for further details). The second column gives the total amount of time (in second) required to compute all the benches (with a timeout of 500s). The last column gives the percentage of time saved using one of the optimality-based method. In the first line, the OBR is used in an unsafe way. Thus, though the result provided by the branch and bound process might be wrong, these experiments underline the potential benefit of the OBR in a branch and bound process. Unfortunately, most of the benefit of OBR is lost by Kearfott's safe approach. A contrario, the constraint-based approach still increases its improvement. The reader may be surprised by the fact that safe constraint-based OBR is even more efficient than the unsafe OBR. Actually, this is due to

[4]See http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html.

[5]Problems with up to 100 variables and 100 constraints are classified as hard problems for rigorous global optimizer in [16].

[6]See http://ylebbah.googlepages.com/icos.

name	LS	PSN	LS+OBR	PSN+OBR
chance	5.67	0.02	7.53	0.03
ex2_1_1	0.40	0.01	0.13	0.01
ex2_1_3	0.52	0.74	0.52	0.83
ex2_1_4	0.94	0.11	0.94	0.09
ex2_1_6	0.66	0.06	2.06	0.07
ex3_1_4	0.76	0.02	0.81	0.02
ex4_1_1	0.15	0.02	1.90	0.02
ex4_1_2	24.91	8.21	24.95	8.39
ex4_1_3	0.10	0.00	0.13	0.01
ex4_1_4	0.08	0.01	0.08	0.01
ex4_1_7	0.08	0.01	0.09	0.00
ex4_1_8	0.08	0.00	0.07	0.00
ex6_1_2	1.44	0.13	1.49	0.13
ex7_2_2	3.69	0.13	3.70	0.17
ex7_2_6	0.80	1.97	0.71	1.66
ex8_1_1	0.46	0.01	0.41	0.01
ex8_1_8	4.12	0.13	4.04	0.16
ex9_1_1	0.53	0.13	0.27	0.13
ex9_1_5	0.29	0.01	0.25	0.00
ex9_2_1	0.67	0.07	0.56	0.07
ex9_2_4	0.59	0.03	0.58	0.03
ex9_2_5	1.60	0.02	1.58	0.02
ex9_2_7	0.50	0.06	0.53	0.06
ex9_2_8	0.10	0.00	0.10	0.00
gbd	0.10	0.02	0.10	0.02
nemhaus	0.11	0.00	0.12	0.00
nvs04	0.23	0.01	0.24	0.01
nvs05	0.02	0.02	0.03	0.03
nvs16	1.42	0.11	0.46	0.11
rbrock	0.11	0.01	0.12	0.00
st_miqp3	0.11	0.00	0.11	0.16
circle	-	0.03	-	0.04
ex2_1_10	-	15.51	-	15.48
ex2_1_2	-	0.06	-	0.05
ex14_1_1	-	0.09	-	0.09
ex14_1_5	-	0.18	-	0.18
ex14_1_6	-	14.02	-	15.93
ex5_2_2_case1	-	0.91	-	0.91
ex9_1_4	-	0.02	-	0.02
ex9_1_6	-	0.53	-	0.42
gear	-	0.00	-	0.00
house	-	0.07	-	0.07
immun	-	3.33	-	3.33
ex14_1_9	-	0.19	1.18	0.19
ex14_1_4	-	-	-	0.10
ex2_1_5	-	-	-	4.77
ex2_1_9	-	23.27	-	-
nvs21	-	25.35	-	-
ex6_1_4	38.78	-	-	-
ex14_2_6	-	-	-	-

Table 2: Timing (in seconds) of different combinations of local search (LS), pseudo-newton (PSN) and optimality-based reduction (OBR). The best times are displayed in blue. A dash (-) indicates the time out occured

the fact that wrong domains reductions achieved by the unsafe OBR prevent the upper-bounding process from improving the current upper bound. Note that the constraint-based approach introduced here is about five time faster than Kearfott's approach.

Table 2 reports the results of our experiments. First column gives the name of the benchmark. The next 4 columns give the time (in seconds) required to solve the benchmarks using the following strategies:

- column 2 (named "LS") uses a local search (Ipopt) to get a feasible point. A Borsuk test is then applied on a box built around this point to prove the existence of a feasible point.
- column 3 (named "PSN") uses the pseudo Newton strategy described above instead of the local search.
- column 4 combines a local search with our safe implementation of the optimality-based reduction.
- column 5 combines the pseudo Newton approach with the safe optimality-based reduction implementation.

The reader may have already note the benefits of our new upper bounding strategy. It is able to solve 14 more benchmarks within the 60s time out than the more usual local search based strategy. Moreover, all the benchmarks but one solved by these two strategies are solved in a much smaller amount of time : "LS" required a total amount of time of 51,24s to solve the first 31 benchmarks while "PSN" has done the same job in 12,07s. Thus, on average, the pseudo Newton strategy is more than four times faster than the usual local search strategy.

The combination of the optimality-based reduction with the different upper bounding strategies slightly improves the performances. For instance, Icos requires 53,83s to solve the 46 first benchmarks with a combination of pseudo Newton and optimality-based reduction while it would have need more than 167,01s to do the same without the help of optimality-based reduction.

6 Conclusion

Constraint programming filtering techniques can be used to implement optimality-based reduction in a simple, safe and efficient way. Thanks to constraint programming, the branch and bound algorithm can take advantage of the OBR through a simple but efficient refutation process. Experiments have shown that our procedure compares well to the Kearfott's procedure. Using constraint-based refutation, OBR is up to five times faster than with Kearfott's procedure. As a result, constraint-based OBR can significantly improve the branch and bound process.

Experiments have also underline the good behavior of the upper bounding detailled in this paper. This new strategy improves drastically the performance of the upper bounding procedure and competes well with a local search.

References

[1] Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming : Theory and Algorithms*. John Wiley & Sons, 1993.

[2] S. L. Campbell and C. D. Jr. Meyer. *Generalized Inverses of Linear Transformations*. New York: Dover, 1991.

[3] J. Cruz and P. Barahona. Global Hull Consistency with Local Search for Continuous Constraint Solving. In *EPIA '01: Proceedings of the10th Portuguese Conference on Artificial Intelligence on Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving*, pages 349–362, 2001.

[4] Andreas Frommer and Bruno Lang. Existence tests for solutions of nonlinear equations using borsuk's theorem. Technical Report BUW-SC 2004/2, Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Wuppertal, M athePrisma, 2004.

[5] Alexandre Goldsztejn, Yahia Lebbah, Claude Michel, and Michel Rueher. Revisiting the upper bounding process in a safe branch and bound algorithm. In *CP2008, 14th International Conference on Principles and Practice of Constraint Programming*, volume 5202 of *Lecture Notes in Computer Science*, pages 598–602. Springer, 2008.

[6] R. Baker Kearfott. Validated probing with linear relaxations. *interval.louisiana.edu/preprints/2005_simplified_feasible_point_verification.pdf*, 2005.

[7] R. Baker Kearfott. Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization. *journal of Optimization Methods and Software*, pages 715–731, October 2006.

[8] Yahia Lebbah. Icos: a branch and bound based solver for rigorous global optimization. *Optimization Methods and Software*, 24(4):709–726, 2009.

[9] Yahia Lebbah and Olivier Lhomme. Accelerating filtering techniques for numeric CSPs. *Artificial Intelligence*, 139(1):109–132, 2002.

[10] Yahia Lebbah, Claude Michel, and Michel Rueher. Using constraint techniques for a safe and fast implementation of optimality-based reduction. In ACM, editor, *Proceedings of SAC'07*, pages 326 – 331, 2007.

[11] Yahia Lebbah, Claude Michel, Michel Rueher, David Daney, and Jean-Pierre Merlet. Efficient and safe global constraints for handling numerical constraint systems. *SIAM Journal on Numerical Analysis*, 42(5):2076–2097, 2004.

[12] Yahia Lebbah, Michel Rueher, and Claude Michel. A global filtering algorithm for handling systems of quadratic equations and inequations. In *CP2002, 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *Lecture Notes in Computer Science*, pages 109–123. Springer, 2002.

[13] Olivier Lhomme. Consistency techniques for numeric CSPs. In *Proceedings of IJCAI'93*, pages 232–238, Chambéry(France), 1993.

[14] Arnold Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004.

[15] Arnold Neumaier and Oleg Shcherbina. Safe bounds in linear and mixed-integer programming. *Mathematical Programming*, pages 283–296, 2004.

[16] Arnold Neumaier, Oleg Shcherbina, Waltraud Huyer, and Tamás Vinkó. A comparison of complete global optimization solvers. *Math. Program.*, 103(2):335–356, 2005.

[17] Michel Rueher, Alexandre Goldsztejn, Yahia Lebbah, and Claude Michel. Constraint programming and safe global optimization. In *2008 International Symposium on Nonlinear Theory and its Applications (NOLTA 2008)*, 2008.

[18] Hong S. Ryoo and Nikolaos V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, pages 107–138, 1996.

[19] P. Van-Hentenryck, D. Mc Allester, and D. Kapur. Solving polynomial systems using branch and prune approach. *SIAM Journal on Numerical Analysis*, pages 34(2):797–827, 1997.